SLOVENSKÁ TECHNICKÁ UNIVERZITA
Fakulta chemickej a potravinárskej technológie
Katedra informatizácie a riadenia procesov
Radlinského 9, 812 37 Bratislava

NÁVRH LQ REGULÁTORA CEZ INTERNET PROSTREDNÍCTVOM PROGRAMU MATLAB

Školiteľ: Ing. Ľuboš Čirka

Vypracoval: Marián Podmajerský

Bratislava 2005

Ďakujem vedúcemu bakalárskej práce Ing. Ľubošovi Čirkovi za pomoc pri získavaní vedomostí z oblasti programovania v HTML kóde a celkovo za vedenie, rady a pripomienky, ktoré mi poskytol pri vypracovaní bakalárskej práce

1 ÚVOD

Vývoj stále nových aplikácií nám umožňuje vypracovanie lepšieho riešenia integrácie odlišných systémov, keďže integrácia je potrebná vo väčšine aplikácií, ktoré sa podieľajú na návrhu riešenia. Prepojením internetovej aplikácie a aplikácie na návrh riešenia sa docieľuje vysoká efektivita, jednoduchosť a prístupnosť. Prostredníctvom webového prehliadača užívateľ zadá vstupné údaje do html formulára, ktoré sa spracujú pomocou naprogramovanej aplikácie na návrh riešenia daného problému pre program, ktorý ju spracuje a vráti údaje do ďalšej html stránky, ktorá sa užívateľovi zobrazí prostredníctvom webového prehliadača.

MATLAB Web Server (MWS) toolbox je prídavný nástroj, ktorý umožňuje spúšťanie vytvorených aplikácií v Matlabe. MWS je ovládaný pomocou vstupného html formulára nachádzajúceho sa na serveri, po ktorého odoslaní MWS spracuje inštrukcie pripojenej aplikácie, nachádzajúcej sa na serveri. Výsledky výpočtov MWS vráti do inej html stránky pomocou číselných údajov, ako sú čísla, vektory, matice, tabuľky, alebo vo forme obrázkov, ktorými sú grafické priebehy simulácií.

Vytvorenie aplikácie pre zadaný výpočet vyžaduje správne porozumenie riešeného problému: teória návrhu riešenia, potrebná simulačná schéma, správne prepojenie html kódu s MWS oboma cestami a znalosť programovacích jazykov.

V tejto práci by som chcel objasniť jednotlivé kroky ku konštrukcii webového modulu pre návrh optimálneho regulátora cez Internet.

2 TEÓRIA NÁVRHU POLYNOMICKÝCH REGULÁTOROV

2.1 Návrh riadenia

Práca je venovaná algebraickým metódam návrhu lineárnych spätnoväzbových regulačných obvodov. Algebraické metódy riadenia je spoločný názov pre metódy analýzy a syntézy dynamických systémov, ktoré vychádzajú z vonkajšieho opisu systému, ktorý chápu ako algebraický objekt. Výsledky spočívajú v riešení algebraických rovníc rôznych typov. Tieto metódy boli pôvodne aplikované na riešenie veľmi jednoduchých problémov riadenia ako stabilizácia a umiestnenie pólov. Postupne sa z nich stali užitočné nástroje na rie1enie širokej triedy úloh (odstránenie poruchy, sledovanie referenčného signálu, optimálne riadenie, robustné riadenie, adaptívne riadenie, atď.).

Zlomkové reprezentácie sú vhodným algebraickým prostriedkom na návrh regulačných obvodov. Podstata ich využitia spočíva v nasledujúcich krokoch:

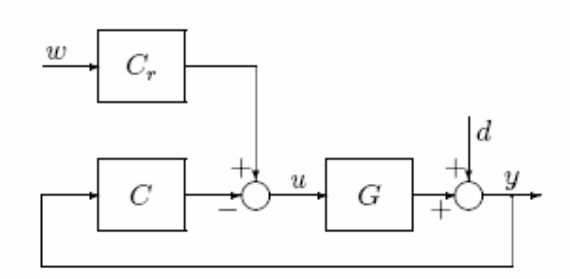
- Najskôr zvolíme základnú požiadavku na regulačný obvod (najčastejšie to býva stabilita).
- Prenos riadeného objektu vyjadríme ako podiel dvoch stabilných prenosov.
- Riešením diofantickej rovnice určíme všetky regulátory, ktoré stabilizujú daný objekt.
- Nakoniec vyhovieme ďalším požiadavkám (napr. optimálnosť, robustnosť) na regulačný obvod tým, že vyberieme vhodný parameter.

Tieto myšlienky boli najskôr rozvinuté v oblasti lineárnych diskrétnych systémov.

V algebraickom zmysle zodpovedajúce označenia rýdzosti môžu byť urobené spôsobom, ktorý je podobný zavedeniu pojmu stability. Návrh regulátorov sa redukuje na riešenie lineárnych diofantických rovníc, ktoré možno v okruhu polynómov algoritmicky riešiť transformáciou na sústavu algebraických rovníc. Tento spôsob syntézy regulátora je výhodný pre adaptívne riadenie, pretože ponúka vyjadrenie parametrov regulátora ako funkciu parametrov prenosu riadeného systému.

V ďalšej časti uvediem výsledky návrhu regulátora metódou umiestnenia pólov, ktorý splňuje určité požiadavky. Jedná sa o návrh, ktorý zaručí stabilitu spätnoväzbového obvodu, asymptotické sledovanie referenčnej (žiadanej) veličiny a odstránenie poruchovej veličiny.

2.1.1 Uzavretý regulačný obvod



Obr.1: Schéma uzavretého regulačného obvodu (2DoF)

Majme uzavretý regulačný obvod podľa obr. 1 opísaný vzťahmi

$y = Gu + d,$

$u = C_r w,$

kde u je akčná veličina, y je regulovaná veličina, w je žiadaná veličina a d je porucha na výstupe riadeného systému.

Riadený systém G, spätnoväzbový regulátor C a priamoväzbový regulátor C_r možno vyjadriť ako podiel dvoch prvkov z R_{ps} alebo P:

- riadený systém G:

$$G = \frac{N_G}{D_G} = \frac{B}{A} \qquad N_G = \frac{B}{M_1}, \quad D_G = \frac{A}{M_1},$$

kde N_G, D_G □ R_{ps} a A, B □ P;

- spätnoväzbový regulátor C:

$$C = \frac{N_C}{D_C} = \frac{Y}{X} \qquad N_C = \frac{Y}{M_2}, \quad D_C = \frac{X}{M_2},$$

kde N_C, D_C □ R_{ps} a X, Y □ P;

- priamoväzbový regulátor Cr:

$$C_r = \frac{N_{C_r}}{D_C} = \frac{R}{X} \qquad N_{C_r} = \frac{Y}{M_2},$$

kde N_{Cr} □ R_{ps} a R □ P;

- žiadaná veličina w:

$$w = \frac{N_w}{D_w} = \frac{H_w}{F_w} \qquad N_w = \frac{H_w}{M_1}, \quad D_w = \frac{F_w}{M_1},$$

kde N_w, D_w □ R_{ps} a H_w, F_w □ P;

- poruchová veličina d:

$$d = \frac{N_d}{D_d} = \frac{H_d}{F_d} \qquad N_d = \frac{H_d}{M_3}, \quad D_d = \frac{F_d}{M_3},$$

kde N_d, D_d □ R_{ps} a H_d, F_d □ P. Z predchádzajúcich vzťahov vyplýva, že M_1, M_2 a M_3 □ S musia byť polynómy s nasledujúcimi stupňami:

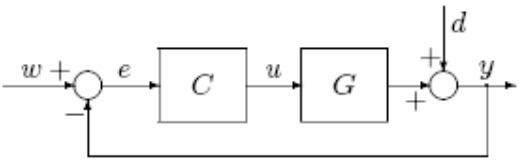
deg $M_1 \geq$ max(deg A, deg F_w), deg $M_2 \geq$ deg X, deg $M_3 \geq$ deg F_d.

Ak $N_{Cr} = N_C$ potom sa jedná o prípad jednoduchého spätnoväzbového riadenia (1 Dof) (obr. 2).

Úlohou syntézy riadenia je určiť racionálne funkcie D_C, N_{Cr} , N_C □ Rps tak, aby uzavretý regulačný obvod bol asymptoticky stabilný a regulačná odchýlka

$$e = w - y = \left(1 - \frac{N_G N_{C_r}}{D_G D_C + N_G N_C}\right) \frac{N_w}{D_w} + \left(\frac{D_G D_C}{D_G D_C + N_G N_C}\right) \frac{N_d}{D_d}$$

konvergovala k nule.



Obr. 2: Schéma uzavretého regulačného obvodu (1DoF)

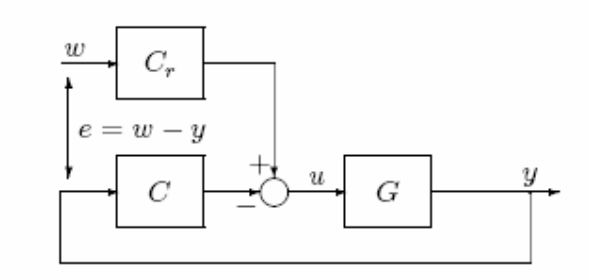
2.1.1.1 Stabilita

Uzavretý regulačný obvod na obr. 1, resp. 2 je stabilný vtedy, keď menovateľ uzavretého regulačného obvodu $D_G D_C + N_G N_C$ je jednotka v okruhu R_{ps}. Z tejto podmienky

vyplýva, že stabilizujúce (spätnoväzbové) regulátory získame riešením diofantickej rovnice (Bezoutovej Identity): $D_G D_C + N_G N_C = 1$.

Ak požadujeme okrem stability uzavretého regulačného obvodu aj stabilitu regulátora, potom hovoríme o silnej stabilizácii.

2.1.1.2 Asymptotické sledovanie



Obr. 3: Sledovanie žiadanej hodnoty

Z požiadavky asymptotického sledovania žiadanej hodnoty (asymptotic tracking) vyplýva, že chyba sledovania (regulačná odchýlka):

$$e = w - y = \left(1 - \frac{N_G N_{C_r}}{D_G D_C + N_G N_C}\right) \frac{N_w}{D_w}$$

musí byť stabilná (obr. 3). Aby odchýlka bola stabilná, musí byť menovateľ jednotka v R_{ps}. To znamená, že musí byť stabilný uzavretý regulačný obvod. Regulačná odchýlka je potom v tvare:

$$e = (1 - N_G N_{C_r}) \frac{N_w}{D_w}.$$

Pretože N_w nie je špecifikované, musí D_w deliť 1 - $N_G N_{Cr}$ v okruhu R_{ps}. Z tejto podmienky vyplýva, že musí existovať prvok N_Z □ R_{ps} taký, že 1 - $N_G N_{Cr} = D_w N_Z$.

Preto priamoväzbový regulátor C_r existuje vtedy a len vtedy, ak D_w a N_G sú nesúdeliteľné v okruhu R_{ps}.

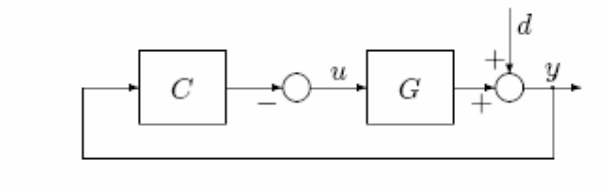
Všetky stabilizujúce 2 DoF (z angl. two degrees of freedom) regulátory zabezpečujúce asymptotické sledovanie žiadanej veličiny sú potom dané nasledovne

$$C = \frac{N_C}{D_C} = \frac{N_{C_0} + D_G Q}{D_{C_0} - N_G Q},$$
$$C_r = \frac{N_{C_r}}{D_C} = \frac{N_{C_{r0}} + D_w Q_r}{D_{C_0} - N_G Q},$$

kde D_{C0} , N_{Cr0} , N_{C0} □ R_{ps} sú partikulárne riešenia a Q, Q_r □ R_{ps} sú ľubovoľné polynómy s obmedzením, že $D_{C0} - N_G Q \neq 0$. Toto obmedzenie nie je veľmi prísne, pretože $D_{C0} - N_G Q$ sa môže identicky zmeniť na nulu iba pre jeden prípad voľby Q. Nakoniec chybu sledovania môžeme vyjadriť v tvare

$$e = N_Z H_w .$$

2.1.1.3 Odstránenie poruchy



Obr. 4: Odstránenie poruchy

Poruchové veličiny z hľadiska získavania informácií sa delia na merateľné a nemerateľné. Štruktúra obvodu pre kompenzáciu porúch (z angl. disturbance rejection) závisí na tom, či je poruchová veličina merateľná alebo nemerateľná.

Merateľná porucha

Ak uvažujeme iba merateľnú poruchovú veličinu, potom jej dynamický účinok na regulovanú veličinu môže byť aproximovaný prenosovou funkciou a na kompenzáciu tejto poruchy sa môže použiť dopredný regulátor.

Nemerateľná porucha

Uvažujme uzavretý regulačný obvod, v ktorom na výstupe systému pôsobí nemerateľná porucha d. Model spätnoväzbového obvodu pre odstránenie poruchy je na obr. 4. Pokiaľ nevieme merať poruchu na výstupe systému, potom ju žiadnym regulátorom nevieme úplne odstrániť. Môžeme však navrhnúť taký regulátor, ktorý zabezpečí, že výstupná veličina bude asymptoticky stabilná. Potom bude vplyv poruchy asymptoticky potláčaný.

Z požiadavky stability uzavretého regulačného obvodu vyplýva, že výstup

$$y = \frac{D_G D_C}{D_G D_C + N_G N_C} \frac{N_d}{D_d}$$

musí byť stabilný. Dosadením bude výstup v tvare

$$y = \frac{D_G D_C N_d}{D_d}$$

Aby výstup bol stabilný, musí byť menovateľ jednotka v R_{ps}. Aby sme to dosiahli, musí sa D_d vykrátiť s čitateľom (za predpokladu, že D_d a D_G sú nesúdeliteľné). Pretože v čitateli je voliteľný iba prvok D_C, môžeme ho zvoliť v tvare $D_C = D_d D_C'$. Stabilizujúce (spätnoväzbové) regulátory, ktoré zabezpečujú asymptotické odstránenie poruchy potom získame riešením modifikovanej diofantickej rovnice

$$D_G D_d \tilde{D}_C + N_G N_C = 1 \qquad (D_C = D_d \tilde{D}_C).$$

Výhoda 2 DoF štruktúry regulátora oproti 1 DoF štruktúre je v tom, že vlastnosti spätnej väzby môžu byť tvarované nezávisle na vlastnostiach sledovania.

2.1.1.4 Metóda umiestnenia pólov

Treba si uvedomiť, že diofantické rovnice nie sú polynomické rovnice, ale rovnice v okruhu R_{ps}. Pretože rovnice v okruhu R_{ps} nevieme riešiť, musíme ich konvertovať na rovnice v okruhu P (t.j. na polynomické rovnice).
Podmienku stability v okruhu polynómov:
$AX + BY = M_1M_2 = M$.

V okruhu polynómov platí aj;

$F_wZ + BR = M_1M_2 = M$:

Pretože M_1 a M_2 sú stabilné polynómy, ich súčin M je tiež stabilný polynóm. Ak polynóm M je zvolený á priori, potom regulátor je navrhnutý metódou umiestnenia pólov. Vnútorná stabilita uzavretého obvodu s pólmi danými polynómom M je dosiahnutá a úloha sledovania má riešenie vtedy a len vtedy, ak dvojice polynómov A, B a F_w, B sú nesúdeliteľné.
Spätnoväzbová časť regulátora je daná riešením diofantickej rovnice

$AX + BY = M,$

a priamoväzbová časť regulátora je daná riešením ďalšej diofantickej rovnice

$F_wZ + BR = M.$

2.2 Polynomické LQ riadenie

Úlohou polynomického optimálneho riadenia je nájsť regulátor, ktorý okrem asymptotickej stability spätnoväzbového systému a minimalizácie kvadratického kritéria zabezpečí aj asymptotické sledovanie referenčnej (žiadanej) veličiny. Pri polynomickom spojitom riadení možno rozlíšiť dva typy kritérií optimálneho riadenia (v kvadratickom zmysle).

Deterministické riadenie (typ LQ - Linear Quadratic) - minimalizuje kritérium

$$J = \frac{1}{2\pi j}\int_{-j\infty}^{j\infty}\left\{u^*(s)\varphi u(s) + y^*(s)\psi y(s)\right\}ds$$

kde $\varphi > 0$, ψ 0 sú váhové koeficienty. Cieľom tohto prístupu je minimalizovať toto kritérium tak, aby uzavretý regulačný obvod (zobrazený na obr. 5) definovaný vzťahmi

$y = Gu + d,\ G = B / A,$

$u = Ce,\ C = Y / X,$

bol asymptoticky stabilný. Potom pravá strana diofantickej rovnice má tvar

$M = D_c D_f$,

kde stabilný polynóm D_c je určený spektrálnym faktorizačným rozkladom

$$D_c^* D_c = \varphi A^* A + \psi B^* B$$

a polynóm D_f je ľubovoľný stabilný polynóm.

Nekonvenčný LQ problém suboptimálneho sledovania je založený na minimalizácii modifikovaného kvadratického funkcionálu

$$J = \int_0^\infty \left(\varphi \tilde{u}^2(t) + \psi e^2(t) \right) dt ,$$

kde e = w - y označuje regulačnú odchýlku. Kvadratický funkcionál môže byť prepísaný použitím Parsevalovho teorému, na získanie výrazu v komplexnej oblasti

$$J = \frac{1}{2\pi j} \int_{-j\infty}^{j\infty} \left\{ \tilde{u}^*(s) \varphi \tilde{u}(s) + e^*(s) \psi e(s) \right\} ds.$$

Nekonvenčné LQ sledovanie pre 1 DoF

Definujme stabilné polynómy D_c a D_f získané zo spektrálnych faktorizácií

$D^*_c D_c = \varphi A^* F^* A F + \psi B^* B$

$D^*_f D_f = A^* H^* A H$

potom stabilita a riešenie deterministickej LQ úlohy je dané polynómami regulátora Y_c, X_c vypočítanými z dvojice diofantických rovníc. Riešenie existuje vtedy ak AF a B sú nesúdeliteľné. Regulátor $C_c = Y_c/(FX_c)$ je daný riešením dvojice diofantických rovníc

$\varphi B^* D_f - AFV^* = D^*_c Y_c$

$\psi A^* F^* D_f - BV^* = D^*_c X_c$

Ak sú polynómy AF a B nesúdeliteľné, potom dvojica diofantických rovníc je redukovaná do diofantickej rovnice

$AFX_c + B_y = D_cD_f$

Nekonvenčné LQ sledovanie pre 2 DoF

Definujme stabilné polynómy D_c a D_f získané zo spektrálnych faktorizácií

$D^*_cD_c = \varphi A^*F^*AF + \psi B^*B$

$D^*_fD_f = A^*H^*AH$

$D^*_rD_r = H^*H$

potom stabilita a riešenie deterministickej LQ úlohy je dané polynómami regulátora Y_c, X_c vypočítanými z dvojice diofantických rovníc. Riešenie existuje vtedy ak AF a B sú nesúdeliteľné. Regulátor $C_c = Y_c/(FX_c)$ je daný riešením dvojice diofantických rovníc

$\varphi B^*D_f - AFV^* = D^*_cY_c$

$\psi A^*F^*D_f - BV^* = D^*_cX_c$

Ak sú polynómy AF a B nesúdeliteľné, potom dvojica diofantických rovníc je redukovaná do diofantickej rovnice

$AFX_c + B_y = D_cD_f$

$FW + BR = D_cD_r$

Výhodou LQ regulátorov je skutočnosť, že regulačné pochody sú optimálne. Toto však z matematického hľadiska platí iba v prípade

- zvoleného kritéria,
- ak objekt a model sú zhodné a lineárne.

3 POUŽITÉ TECHNOLÓGIE

Pre správny chod MWS je potrebné splniť podmienky zo strany užívateľa aj najmä serveru:

Strana klienta vyžaduje:

- TCP/IP pripojenie počítača na sieť
  Prenosová rýchlosť závisí na požadovanom komforte. Veľkosť dát, ktorá môže server vrátiť (ako odpoveď na jeden dotaz) je obmedzená na 256 kB.
- program pre prezeranie HTML stránok

V súčasnosti najbežnejšie používané prehliadače sú Netscape Navigator, Microsoft Internet Explorer, Opera, Mozilla. Nutnosť použitia vyšších verzií vyššie menovaných prehliadačov závisí na tom, či budú do formulárových a výstupných HTML stránok vkladané novšie značky, Java Skripty a podobne.

Strana serveru vyžaduje:

- počítač pripojený na sieť (TCP/IP)

nároky na operačnú pamäť: súčet nárokov jednotlivých programov + max 256 kB na jedného užívateľa (klienta)

operačný systém Windows NT nebo LINUX

program Matlab 5.3 alebo vyšší a jeho súčasť MWS

WWW server, ktorý podporuje CGI – Apache

3.1 Web Server – Apache

Apache je nakonfigurovaný WWW server, ktorý dokáže spúšťať skripty (či iné programy) s možnosťou predávať dáta zaslané od vzdialených užívateľov prostredníctvom CGI rozhrania. Je to nevyhnutná požiadavka na počítač, ktorý má slúžiť ako server. Apache je nenáročný ľahko dostupný WWW Server pracujúci na platforme Windows i Linux. Pri jeho inštalácii vyžaduje inštalačný program niekoľko informácií: patrí medzi ne meno domény či

jej IP adresa, port, adresár kam sa majú uložiť programové a konfigurační súbory, kmeňový adresár pre WWW stránky a emailová adresa správcu serveru. Jedným z dôležitých dotazov je to, ako sa má WWW server Apache spúšťať. Je potrebné zvoliť voľbu service, aby server pracoval i pri odhlásení užívateľa (správcu). Po dokončení inštalácie by mal byť už server funkčný. Desiatky ďalších parametrov je možné ovplyvniť modifikáciou parametrov uložených v niekoľkých konfiguračných súboroch v podadresári conf. Takto sa uskutočňuje nastavenie jednotlivých vlastností WWW serveru. Najdôležitejšie pre vzdialený prístup k MWS je nastavenie niekoľkých parametrov, ktoré vo windowse nastavíme v súbore http.conf nachádzajúceho sa v adresári apacha /Conf/:

- povolenie behu CGI aplikácií a nastavenie adresára, z ktorého budú spúšťané

AddModule mod_cgi.c

ScriptAlias /cgi-bin/ "C:/Apache/cgi-bin/"

- nastavenie adresára, v ktorom budú uložené súbory a nastavenie prístupu k adresáru

DocumentRoot "C:/www"

<Directory />

Options FollowSymLinks

AllowOverride None

</Directory>

- do adresára cgi-bin/ je potrebné nakopírovať spúšťací súbor MWS – matweb.exe a vytvoriť súbor matweb.conf s nasledovnou štruktúrou:
- *do hranatých zátvoriek uvedieme názov našej aplikácie, napr. [LQreg]*
- *ďalej mlserver = a názov nášho serveru, napr. mlserver = localhost*
- *nakoniec uvedieme za príkazom mldir = cestu k našej apliácii, napr. mldir = C:/www/LQreg*
- celý príklad:

[LQreg]

mlserver=localhost

mldir=C:\www\LQreg

Týmto je WWW server pripravený na spoluprácu s MWS.

3.2 MATLAB Web Server (MWS)

Keď máme MWS nainštalovaný a nakonfigurovaný môžeme začať s tvorbou prezentácie. Od tvorcov prezentácie sa žiada, aby pripravil:

- HTML stránku s formulárom pre zadanie parametrov výpočtu
- funkciu (funkcia) v Matlabe (m-file) pre vlastný výpočet (napr. LQreg.m).

Parametrom tejto funkcie je štruktúra so vstupnými dátami (z formulára). Výstupom tejto funkcie je textový reťazec s výslednou HTML stránkou (postupnosť znakov:

<html><head> hlavička </head><body> obsah stránky </body></html>)

- šablónu výstupnej stránky HTML pre jednoduché generovanie skutočnej výstupnej stránky HTML (napr. sablona.html).

MWS disponuje funkciou, ktorá vie jednoduchým postupom nahradiť parametre v šablóne skutočnými údajmi (číslo, textový reťazec, matice čísiel ako tabuľka, meno súboru s obrazkom grafu, ...). Nie je teda nutné zaoberať sa vo funkcii LQreg.m generovaním nepremnných častí HTML kódu. Tento súbor však nieje nevyhnutný, pretože kód výslednej HTML stránky môže byť generovaný aj priamo funkciou LQreg.m.

Vlastný proces výmeny dát prebieha následovne. Po zobrazení stránky formular.html zadá užívateľ požadované parametre a odošle formulár stlačením tlačidla "Odoslať" (angl. Submit). Údaje sa prenesú cez rozhranie CGI metódou POST programu matweb.exe na WWW server. Časť kódu v súbore formular.html vyzerá takto:

```
...
<form action="/cgi-bin/matweb.exe" method="post" name="formular"">
<input type="hidden" name="mlmfile" value="LQreg">
...užitočné dáta...
</form>
…
```

Medzi údajmi formulára sa pomocou skrytého poľa *mlmfile* prenesie aj meno funkcie (bez koncovky "m"), ktorá bude spracovávať údaje. Program *matweb.exe* je klientom MWS. Prijaté údaje transformuje a vracia späť. Matlab Web Server potom spustí prislúchajúcu funkciu (LQreg.m).

Teraz už spustená funkcia (LQreg.m) vytvára zo vstupných dát údaje, ktoré sa majú objaviť vo výslednej HTML stránke. Aby sa maximálne zjednodušilo zostavovanie ich kódu,

disponuje Matlab Web Server funkciou *htmlrep*. Do nej vstúpi ako parameter názov súboru so šablónou výstupnej HTML stránky (sablona.html) a štruktúra s vypočítanými údajmi (čísla, vektory, matice, textové reťazce). Funkcia načíta túto šablónu a nahradzuje výskyty $premenná$ skutočnými hodnotami (čísla, reťazce). Vektory a matice sa automaticky formátujú do podoby tabuliek (vkladajú sa značky <TABLE>, <TR>, <TD>, ...). Výsledkom tejto funkcie je reťazec s výsledným kódom HTML stránky. Tento reťazec sa s ukončením behu funkcie (LQreg.m) vráti cez WWW server prehliadačov HTML stránok. (pozn. funkcia *htmlrep* sa volá vo vnútri funkcie, ktorá spracováva dáta (LQreg.m)).

3.3 HTML

HTML je jazyk slúžiaci na programovanie www stránok. Umožňuje nám jednoducho vytvoriť vstupný formulár formular.html a výstupnú šablónu sablona.html.
Obsah HTML stránky je uzavretý medzi tagmi <HTML> a </HTML>, v ktorých sú vnorené ďalšie tagy <BODY> a </BODY>.

HTML jazyk obsahuje množstvo ďalších príkazov = tagov, umožňujúcich vytvoriť jednoduchý a účelný vzhľad www stránky.

Párový tag <TABLE></TABLE> nám vytvorí tabuľku, ktorej počet a riadkov a stĺpcov si určujeme sami ďalšími tagmi: <tr> a </tr> pre riadky a do nich sa vnárajú tagy <td> a </td> pre stĺpce. Tag <TABLE> obsahuje nasledujúce parametre:

- align – obtekanie tabuľky ostatným textom
- cellpadding - vnútorný okraj buniek
- cellspacing - vonkajší okraj buniek
- border - šírka rámčeka buniek
- width - minimálna šírka tabuľky
- height - minimálna výška
- background - obrázok na pozadí buniek
- bgcolor - farba pozadia buniek
- bordercolor - farba rámčeka
- frame vykreslenie rámčeku okolo príkazmi: void, border, box, hsides, vsides, above, below, lhs, rhs
- rules vykreslení mriežky príkazmi none, all, rows, cols, groups

Párový tag slúži na tvorbu odsekov textu.

Najdôležitejšie HTML tagy v mojej práci, sú tie ktoré sú potrebné na vytvorenie formulára. Preto ich popíšem osobitne v nasledujúcej časti.

3.3.1 Formuláre

Začiatok formulára označuje tag <FORM>, koniec </FORM>. Medzi ne sa vkladajú jednotlivé tagy ovládacích prvkov (textové pole, radio button, checkbox, ...). Tag FORM má nasledujúce parametre:

- action - adresa (URL), na ktorej je umiestnený obslužný program. Pre nás nim bude vždy matweb.exe
- method - určuje spôsob zasielania formulárových dát. Použijeme metódu POST.

Príklad prázdneho formulára:

```
<form action="/cgi-bin/matweb.exe" method="POST">
</form>
```

Teraz je potrebné vložiť skryté pole s menom mlmfile, jeho pred nastavený údaj určuje, ktorá funkcia (m-file) sa má spustiť (bez prípony m).

```
<INPUT type="hidden" name="mlmfile" value="vypocet">
```

Nepárová tag <INPUT> sa používa aj pre ďalšie vstupné ovládacie prvky. Význam jej nasledujúci:

- type - určuje druh
- name - názov
- value – prednastavená hodnota (po načítaní stránky alebo po stlačení tlačidla reset)

Parameter *type* môže nadobúdať tieto hodnoty:

- text - vloží jednoriadkové textové vstupné pole formulára. Ak je určená počiatočná hodnota, zobrazí sa v textovom poli. Veľkosť zobrazovaného poľa (počet znakov) možno určiť parametrom SIZE; maximálny počet znakov pomocou parametra MAXLENGTH.

Hodnotou tohto prvku je text zadaný v textovom poli.

- password - vloží jednoriadkové textové pole na zadanie hesla. Namiesto znakov zadávaných užívateľom sa zobrazuje znak "*" (hviezdička). Veľkosť zobrazovaného

poľa (počet znakov) možno určiť parametrom SIZE; maximálny počet znakov pomocou parametra MAXLENGTH.

Hodnotou tohto prvku je text zadaný v textovom poli. Pozor! zadávanie hesla pomocou formulára nie je bezpečné a to najmä pri použití metódy "GET"!

- checkbox - vytvorí zaškrtávacie políčko formulára. Počiatočné "zaškrtnutie" možno dosiahnuť použitím parametra CHECKED.

Hodnotou zaškrtávacieho políčka je hodnota "on", ak je zaškrtnutý.

- radio - vytvorí prepínač. Prepínač sa skladá zo skupiny vstupných polí typu "radio", ktoré majú rovnaké meno a z ktorých môže byt aktívny práve jeden. Počiatočné "zaškrtnutie" možno dosiahnuť použitím parametra CHECKED v niektorom prepínači.

Hodnotou prepínača je hodnota uvedená v parametri VALUE toho prepínača, ktorý si užívateľ vybral.

- file - vytvorí "browsovacie" tlačidlo a textové pole (na pripadne zadanie mena súboru) na vyber súboru na lokálnom počítači a umožní odoslanie súboru spolu s formulárom. Pozri <FORM> a parameter ENCTYPE.
- hidden - umožňuje vložiť do formulára skrytú informáciu, ktorá sa bude prenášať pri každom odoslaní formulára. Položka sa nijakým spôsobom nezobrazuje v prehliadači.

Hodnotou skrytého poľa je priamo hodnota uvedená v parametri VALUE.

- submit - vloží odosielacie tlačidlo formulára. Aktivovaním tohto tlačidla sa odovzdá riadenie programu, ktorý je uvedený v parametri ACTION príkazu <FORM>.

Tento prvok zároveň vracia po odoslaní hodnotu uvedenú v VALUE.

- reset - vygeneruje nanovo formulár s počiatočnými hodnotami.

Tento prvok nevracia hodnotu.

3.3.2 JavaScript

JavaScript je programovací jazyk, ktorý se používa v internetových stránkach. Zapisuje se priamo do HTML kódu, čo je veľká výhoda, pretože je to jednoduché.

JavaScript je klientský skript. To znamená, že sa program odosiela se stránkou na klienta (do prohliedača) a až potom tam je vykonávaný. (Protikladom klientských skriptov sú skripty serverové, ktoré sä vykonávané na serveri a ku klientovi idú už len výsledky.)

čo je treba vedieť

- HTML, aspoň základy
- základy programovania, aspoň syntax

charakteristiky jazyka - JavaScript je jazyk

- interpretovaný -- nemusí sa kompilovať
- objektový -- využíva objektov prehliadača a zabudovaných objektov
- závislý na prehliadači -- funguje ale vo väčšine prehliadačov
- case sensitivný -- záleží na veľkosti písma v zápise
- syntaxou podobný jazykom C, Java a podobným

obmedzenie jazyka

- JavaScript funguje len v prehliadači.
- Užívateľ môže JavaScript zakázať
- Existujú rôzne odlišné verzie jazyka i prehliadačov, čo vedie k častým chybám.
- Nevie pristupovať k súborom (okrem cookies) ani k žiadnym systémovým objektom.
- Nevie žiadne dáta uložiť (okrem cookies).

požitie v HTML

- možno kontrolovať odchádzajúce formuláre, či sú vyplnené a ak sú či sú vyplnené správnymi požadovanými údajmi
- ovládanie prvkov a objektov HTML stránky
- píše sa do prúdu dokumentu:

```
...
Toto je normálny text stránky.<br>
<script>
document.write("Napísal JavaScript");
</script>
...
```

4 TVORBA PREZENTÁCIÍ V MWS

4.1 Všeobecný popis

Štruktúra s dátami formulára je odovzdaná funkcii určenej vo formulárovom dotaze (ako skrytý parameter). Táto funkcia vracia formou výstupného reťazca celý kód výslednej HTML stránky spolu s vypočítanými údajmi, pričom ak sa chceme vyhnúť vypisovaniu jednotlivých vstupných ale i pomocných výpočtových premenných, je potrebné jednotlivé riadky pri písaní funkcie ukončovať bodkočiarkou. Výpis je totiž presmerovaný na klientský WWW prehliadač (štruktúra HTML dokumentu by potom nebola správna).

Štruktúra m-súboru, ktorá spracováva vstupné dáta z predošlého formulára a vypočíta ich má tieto kroky:

- function retstr = regul(instruct, outfile) - definovanie funkcie
- cd(instruct.mldir); - nastavenie pracovného adresára na hodnotu uloženú v položke *mldir*, táto položka sa pridáva automaticky a jej hodnota je určená nastavením v súbore matweb.conf
- nasledujú príkazy matlabu, napr. a = 12;
- outstruct.premenna = 12; - pridelí premennú, ktorá bude výsledkom výpočtov m-súboru
- templatefile = which('template.html'); - nastavenie súboru, ktorý sa spustí v prípade hlásenia chyby počas behu prezentácie
- retstr = htmlrep(outstruct, templatefile); - funkcia vracajúca výsledné premenné MWS

4.2 Aplikovanie na konkrétny prípad

4.2.1 Vstupný formulár

Doteraz sme videli len všeobecný zápis formulára potrebný pre správny beh MWS. Teraz si ukážeme ako by mal vyzerať hotový formulár:

```
<FORM      action="/cgi-bin/matweb.exe"      method="post"      name="formular"
onSubmit="return potvrd();">

<!-- Zadefinovanie formulára a konfigurácia -->

   <input type="hidden" name="mlmfile" value="LQreg">

Čas štartu simulácie:
```

```
<input type="text" size="10" maxlength="10" name="cas_simulacie" >

Čas dosiahnutia žiadanej veličiny:

<input type="text" size="10" maxlength="10" name="cas_ziadanej_hodnoty" >

Veľkosť žiadanej veličiny:

<input type="text" size="10" maxlength="10" name="velkost_ziadanej_hodnoty" >

Čas zasiahnutia poruchy:

<input type="text" size="10" maxlength="10" name="cas_poruchy" >

Veľkosť poruchy:

<input type="text" size="10" maxlength="10" name="velkost_poruchy" >

Čitateľ prenosu systému:

<input type="text" size="32" maxlength="30" name="citatel_prenosu" >

Menovateľ prenosu systému:

<input type="text" size="32" maxlength="30" name="menovatel_prenosu" >

Váhový koeficient fi

<input type="text" size="10" maxlength="10" name="fi">

Váhový koeficient psi

<input type="text" size="10" maxlength="10" name="psi">

<input type="submit" name="Submit" value="simulovať" >

<!-- Tlačidlo simulovať odošle formulár na spracovanie, najprv pre JavaScript
potom MWS -->

<input type="reset" name="Reset" value="zmazať">

<!-- Tlačidlo reset zmaže údaje vo formulári -->

<input type="button" name="ukazka" value="naplniť ukážkou" onclick="napln();">

<!-- Tlačidlo naplniť ukážkou naplní formulár ukážkou -->

   </FORM>
```

Overovanie správnosti vyplnenia formuláru prebieha JavaScriptom, tak že najprv overíme, či je zadané číslo, v prípade polynómov otestujeme prvý a posledný znak ak sú to [ ], tak rozdelením vnútra podľa medzier získame čísla, ktoré znovu otestujeme či sú naozaj čísla:

```
<!-- Naplnenie ukážkou -->
```

```
function napln()
{

document.formular.cas_simulacie.value="500";
document.formular.cas_ziadanej_hodnoty.value="10";
document.formular.velkost_ziadanej_hodnoty.value="5";
document.formular.cas_poruchy.value="100";
document.formular.velkost_poruchy.value="0.2";
document.formular.citatel_prenosu.value="[1 3 4]";
document.formular.menovatel_prenosu.value="[2 3 5 7]";
document.formular.fi.value="100";
document.formular.psi.value="2";
}

    <!-- Overenie vyplnenia formulára a správneho vyplnenia formulára -->

function potvrd(){
var chyba = false; //zadefinovanie premennej, ktorá neodosle formular
       if  (document.formular.cas_simulacie.value=='')  {alert('Zadajte  prosím  dĺžku  času
simulácie!');chyba='true';} // zistuje ci je zadana hodnota
       if  (isNaN(parseInt(document.formular.cas_simulacie.value)))  {alert('Zadajte  prosím
dĺžku času simulácie ako číslo nie znaky!');chyba='true';}  //zistuje ci zadana hodnota je
cislo
       if  (document.formular.cas_ziadanej_hodnoty.value=='')  {alert('Zadajte  prosím  čas
žiadanej hodnoty!');chyba='true';} // zistuje ci je zadana hodnota
       if   (isNaN(parseInt(document.formular.cas_ziadanej_hodnoty.value)))   {alert('Zadajte
prosím čas žiadanej hodnoty ako číslo nie znaky!');chyba='true';} //zistuje ci zadana hodnota
je cislo
       if   (document.formular.velkost_ziadanej_hodnoty.value=='')   {alert('Zadajte   prosím
veľkosť žiadanej hodnoty!');chyba='true';} // zistuje ci je zadana hodnota
       if  (isNaN(parseInt(document.formular.velkost_ziadanej_hodnoty.value))) {alert('Zadajte
prosím veľkosť žiadanej hodnoty ako číslo nie znaky!');chyba='true';} //zistuje ci zadana
hodnota je cislo
       if     (document.formular.cas_poruchy.value=='')     {alert('Zadajte     prosím     čas
poruchy!');chyba='true';} // zistuje ci je zadana hodnota
       if  (isNaN(parseInt(document.formular.cas_poruchy.value))) {alert('Zadajte prosím čas
poruchy ako číslo nie znaky!');chyba='true';} //zistuje ci zadana hodnota je cislo
       if  (document.formular.velkost_poruchy.value=='')  {alert('Zadajte  prosím  veľkosť
poruchy!');chyba='true';} // zistuje ci je zadana hodnota
       if  (isNaN(parseInt(document.formular.velkost_poruchy.value))) {alert('Zadajte prosím
veľkosť poruchy ako číslo nie znaky!');chyba='true';} //zistuje ci zadana hodnota je cislo
       if  (document.formular.citatel_prenosu.value=='')  {alert('Zadajte  prosím  čitateľa
prenosu!');chyba='true';} // zistuje ci je zadana hodnota
var t=document.formular.citatel_prenosu.value; //nacita hodnotu
var c1,c2,hodnota; //pomocne premenne
c1=t.charAt(0);
       if (c1!='[') chyba='true'; //skontroluje ci je to prenos
c2=t.charAt(t.length-1);
       if (c2!=']') chyba='true';
hodnota=t.substring(1,t.length-1).split(" "); //vytiahne cisla medzi []
for(i = 0; i < hodnota.length; i++){
```

```
	if (isNaN(parseInt(hodnota[i]))) {alert('Zadajte prosím medzi zátvorky čísla miesto znakov!');chyba='true';} //zistuje ci zadana hodnota je cislo
}
	if (document.formular.menovatel_prenosu.value=='') {alert('Zadajte prosím menovateľa prenosu!');chyba='true';} // zistuje ci je zadana hodnota
var t=document.formular.menovatel_prenosu.value;
var c1,c2,hodnota;
c1=t.charAt(0);
	if (c1!='[') chyba='true'; //skontroluje ci je to prenos
c2=t.charAt(t.length-1);
	if (c2!=']') chyba='true';
hodnota=t.substring(1,t.length-1).split(" ");//vytiahne cisla medzi []
for(i = 0; i < hodnota.length; i++){
	if (isNaN(parseInt(hodnota[i]))) {alert('Zadajte prosím medzi zátvorky čísla miesto znakov!');chyba='true';} //zistuje ci zadana hodnota je cislo
}
	if (document.formular.fi.value=='') {alert('Zadajte prosím koeficient fi!');chyba='true';} // zistuje ci je zadana hodnota
	if (isNaN(parseInt(document.formular.fi.value))) {alert('Zadajte prosím koeficient fi ako číslo nie znaky!');chyba='true';} //zistuje ci zadana hodnota je cislo
	if (document.formular.psi.value=='') {alert('Zadajte prosím koeficient psi!');chyba='true';} // zistuje ci je zadana hodnota
	if (isNaN(parseInt(document.formular.psi.value))) {alert('Zadajte prosím koeficient psi ako číslo nie znaky!');chyba='true';} //zistuje ci zadana hodnota je cislo
	if (chyba) {return false;} //ak nastane niekde chyba neodosle formular
	if (chyba==false) document.formular.submit(); //formular bez chyby, ide na spracovanie MWS

}
```

4.2.2 Spracovanie údajov v Matlabe

Takto vyzerá náš súbor LQreg.m

```
function retstr = LQreg(instruct, outfile) //začiatok m-suboru
// kontrola či prišli údaje z formulára
if (isempty(instruct.citatel_prenosu) |isempty(instruct.menovatel_prenosu))

outstruct.errormsg='Nezadali ste niektory z polynomov prenosu systemu!'; errorcode=1;

elseif (isempty(instruct.fi)| isempty(instruct.psi))
	outstruct.errormsg='Nezadali ste niektory vahovy koeficient!'; errorcode=1;

elseif	(isempty(instruct.cas_simulacie)|	isempty(instruct.cas_ziadanej_hodnoty)| isempty(instruct.velkost_ziadanej_hodnoty)|	isempty(instruct.cas_poruchy)| isempty(instruct.velkost_poruchy))
	outstruct.errormsg='Nezadali ste niektory z parametrov simulacie!'; errorcode=1;

elseif (spolycheck(instruct.citatel_prenosu) | spolycheck(instruct.menovatel_prenosu)),
	outstruct.errormsg='Nespravne zadane polynomy. Pravdepodobne Vam niekde chybaju hranate zatvorky, alebo ste zadali neciselnu hodnotu';
	errorcode=2;
```

```
elseif (spolydeg(instruct.citatel_prenosu) > spolydeg(instruct.menovatel_prenosu)),
       outstruct.errormsg='Polynom citatela nesmie byt vyssieho radu ako polynom menovatela!';
   errorcode=4;

else
       errorcode=0;
end
//ak dôjde k chybe spustí sa náhradná stránka
if errorcode~=0,
       templatefile = which('template.html');
       retstr = htmlrep(outstruct, templatefile);
       return
end
//konverzia na polynomy a cisla
B=mat2pol(str2num(instruct.citatel_prenosu));
A=mat2pol(str2num(instruct.menovatel_prenosu));
Fi=str2num(instruct.fi);
Psi=str2num(instruct.psi);
F=s;
//spektrálna faktorizácia
[Df,j]=spf(A'*A);
[Dr,j]=spf(1'*1);
[Dc,j]=spf(Fi*A'*F'*A*F+Psi*B'*B);
[P,Q]=axbyc(A*F,B,Dc*Df);
[W,R]=axbyc(F,B,Dc*Dr); //spektrálna faktorizácia len pre 2DoF
//konverzia
P1=pol2mat(P);
Q1=pol2mat(Q);
B1=pol2mat(B);
A1=pol2mat(A);
W1=pol2mat(W); //len pre 2DoF
R1=pol2mat(R); //len pre 2DoF
scheme_name='LQregs2';
open_system('LQregs2'); //otvorenie schémy
//úprava prenosov
gG=sprintf('%s/G',scheme_name);
gR=sprintf('%s/R',scheme_name);
gR1=sprintf('%s/R1',scheme_name);
//vyplnenie simulačnej schémy
AA=sprintf('[%s]',num2str(A1));
BB=sprintf('[%s]',num2str(B1));
set_param(gG,'Denominator',AA);
set_param(gG,'Numerator',BB);
QQ=sprintf('[%s]',num2str(Q1));
PP=sprintf('[%s]',num2str(P1));
set_param(gR,'Denominator',PP);
set_param(gR,'Numerator',QQ);
//len pre 2DoF
WW=sprintf('[%s]',num2str(W1));
```

```
RR=sprintf('[%s]',num2str(R1));
set_param(gR1,'Denominator',WW);
set_param(gR1,'Numerator',RR);
//nastavenie parametrov simulácie
cas_sim = num2str(double(str2num(instruct.cas_simulacie)));
set_param(scheme_name,'Solver','ode45','StopTime',cas_sim);
set_param(scheme_name,'Solver','ode45','Max step size','auto');
cas_ziad = num2str(double(str2num(instruct.cas_ziadanej_hodnoty)));
cas_chyba = num2str(double(str2num(instruct.cas_poruchy)));
big_ziad = num2str(double(str2num(instruct.velkost_ziadanej_hodnoty)));
big_chyba = num2str(double(str2num(instruct.velkost_poruchy)));
ziadana=sprintf('%s/ziad',scheme_name);
set_param(ziadana,'Time',cas_ziad);
set_param(ziadana,'Before','0');
set_param(ziadana,'After',big_ziad);
porucha=sprintf('%s/poruch',scheme_name);
set_param(porucha,'Time',cas_chyba);
set_param(porucha,'Before','0');
set_param(porucha,'After',big_chyba);
warning off;
sim(scheme_name);
bdclose('all');
//nastavenie menoviek simulácie
f=figure('visible','off');
plot(simout(:,1),simout(:,2));
xlabel('t [s]'); ylabel('w(t)');
titlestring=sprintf('Vysledok simulacie vypocitaneho regulatora');
title(titlestring);
outstruct.GraphFileName = sprintf('priebeh.jpeg');
wsprintjpeg(f, outstruct.GraphFileName); //výstup obrázku zo simulácie
close all;
//zadanie výstupných veličín
outstruct.cas_simulacie=instruct.cas_simulacie;
outstruct.cas_ziadanej_hodnoty=instruct.cas_ziadanej_hodnoty;
outstruct.velkost_ziadanej_hodnoty=instruct.velkost_ziadanej_hodnoty;
outstruct.cas_poruchy=instruct.cas_poruchy;
outstruct.velkost_poruchy=instruct.velkost_poruchy;
outstruct.citatel_prenosu=pol2html(B1);
outstruct.menovatel_prenosu=pol2html(A1);
outstruct.fi=instruct.fi;
outstruct.psi=instruct.psi;
outstruct.r1_citatel=pol2html(Q1);
outstruct.r1_menovatel=pol2html(P1);
outstruct.r2_citatel=pol2html(W1);
outstruct.r2_menovatel=pol2html(R1);
//koniec m-suboru
templatefile = which('sablona.html');
if (nargin == 1)
   retstr = htmlrep(outstruct, templatefile);
elseif (nargin == 2)
   retstr = htmlrep(outstruct, templatefile, outfile);end
```

4.2.3 Výstupná stránka

Šablóna sa líši od bežnej HTML stránky tým, že na zvolených miestach sú zapísané medzi znakmi $ mená položiek – premenných, ktoré majú byť nahradené konkrétnymi hodnotami tak, ako ich predáva daná obslužná funkcia (v tomto prípade *vypocet*):

```
Vysledok= $meno_premennej$
```

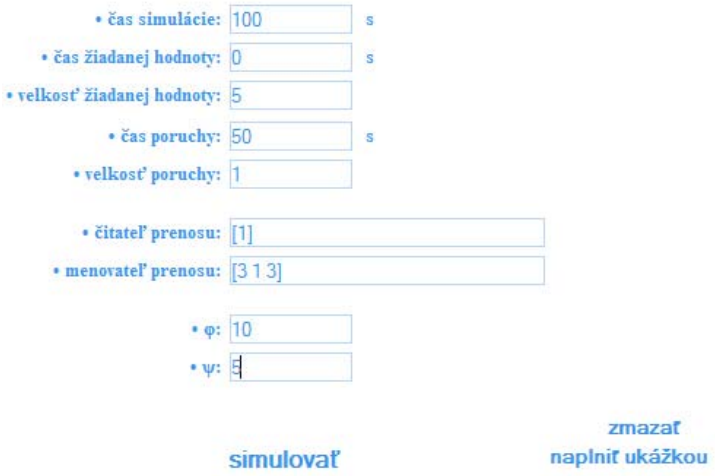
V prípade, že výstup má charakter tabuľky alebo matice a ak chceme, aby sa matica zobrazila ako tabuľka (s orámovaním), využijeme nasledujúci kód. Počet stĺpcov a riadkov je pritom určený skutočnou veľkosťou matice. Príkazy <*tr*> a <*td*> možno doplniť parametrami, aby sme docielili požadovaný vzhľad (zarovnanie, farby, orámovanie,...). To isté je možné upraviť aj priamo v príkaze <*table*>.

```
<table autogenerate="$meno_matice$">
<tr>
<td>
</td>
</tr>
</table>
```

5 PRÍKLADY

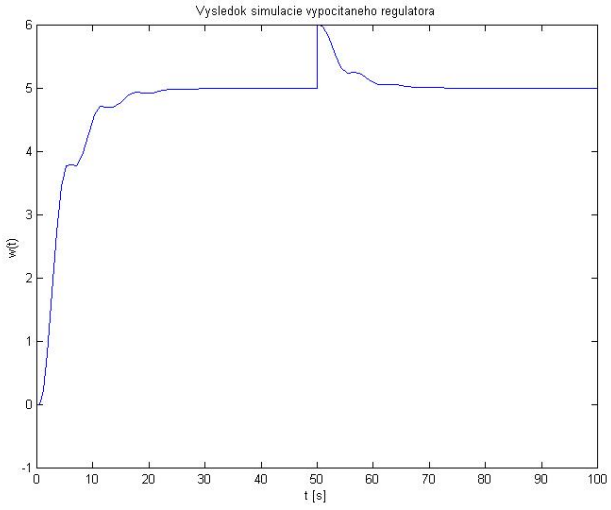
5.1 1 DoF

Po otvorení súboru formular.html nájdeme formulár na zadávanie parametrov. Zadávame od vrchu dĺžku simulácie, počiatočný čas žiadanej veličiny, konečná veľkosť žiadanej veličiny, čas, v ktorom sa objaví porucha, veľkosť tejto poruchy, čitateľa a menovateľa prenosu nášho spätnoväzbového modelu a váhové keoficienty.

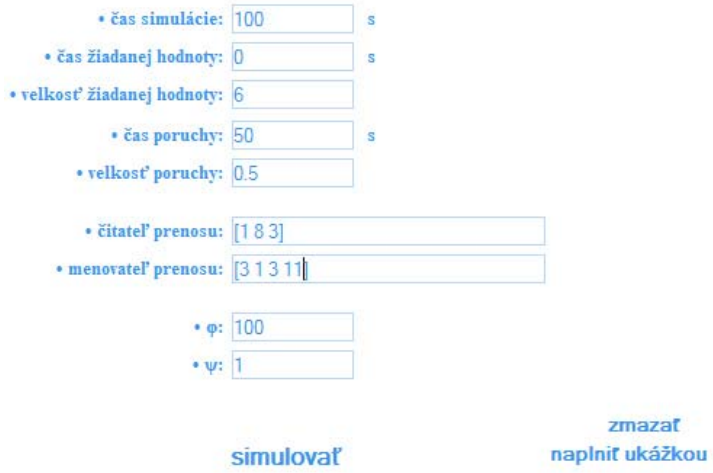


Ako je z obrázku vidieť dĺžka simulácie je 500 s, počiatočný čas žiadanej veličiny 0 s, konečná veľkosť žiadanej veličiny 5, čas, v ktorom sa objaví porucha 50 s, veľkosť tejto poruchy 1, čitateľa a menovateľa prenosu nášho spätnoväzbového modelu a váhové keoficienty.

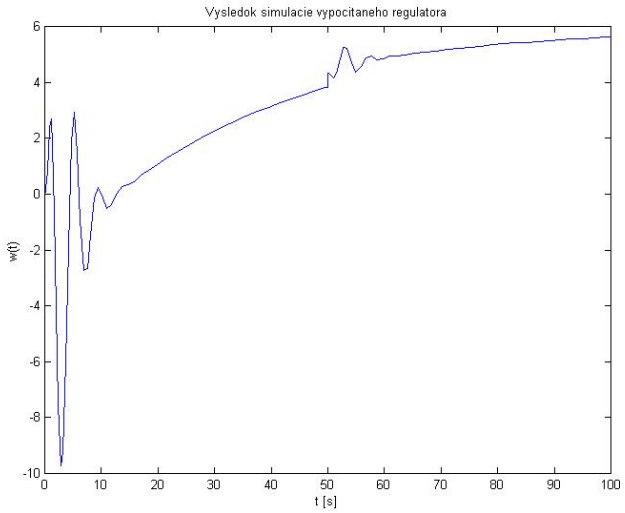
Výsledok po odoslaní je



Druhý krát som si zvolil dĺžku simulácie 500 s, počiatočný čas žiadanej veličiny 0 s, konečná veľkosť žiadanej veličiny 6, čas, v ktorom sa objaví porucha 50 s, veľkosť tejto poruchy 0.5, iný čitateľ a menovateľ prenosu nášho spätnoväzbového modelu i iné váhové keoficienty

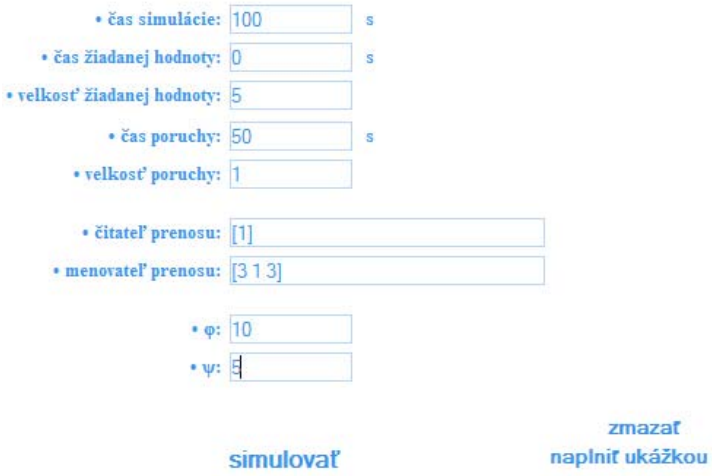


a výsledok je



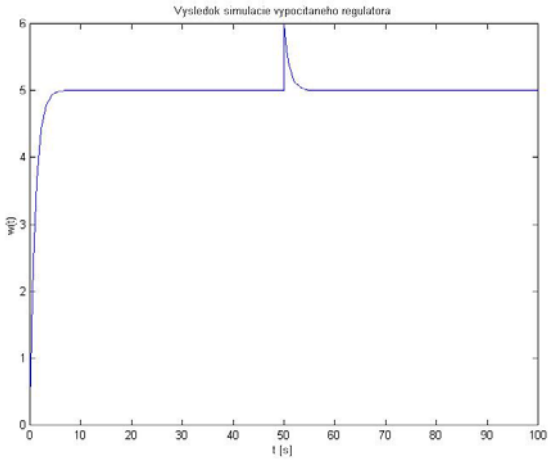
5.2 2 DoF

Pre 2 DoF obvod sú priebehy odlišné.

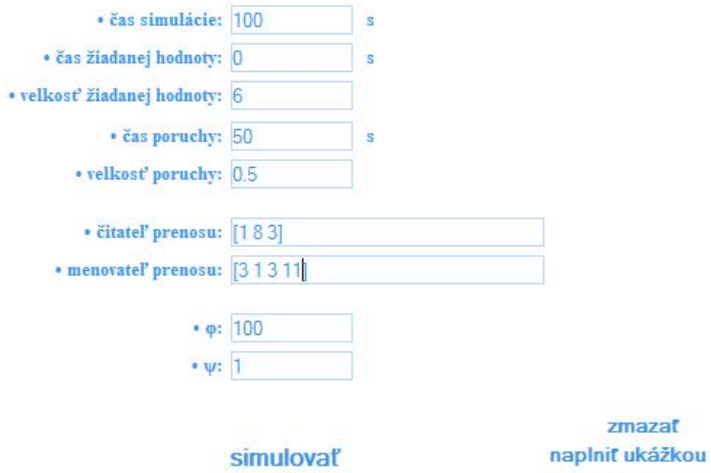


Ako je z obrázku vidieť dĺžka simulácie je 500 s, počiatočný čas žiadanej veličiny 0 s, konečná veľkosť žiadanej veličiny 5, čas, v ktorom sa objaví porucha 50 s, veľkosť tejto poruchy 1, čitateľa a menovateľa prenosu nášho spätnoväzbového modelu a váhové keoficienty.

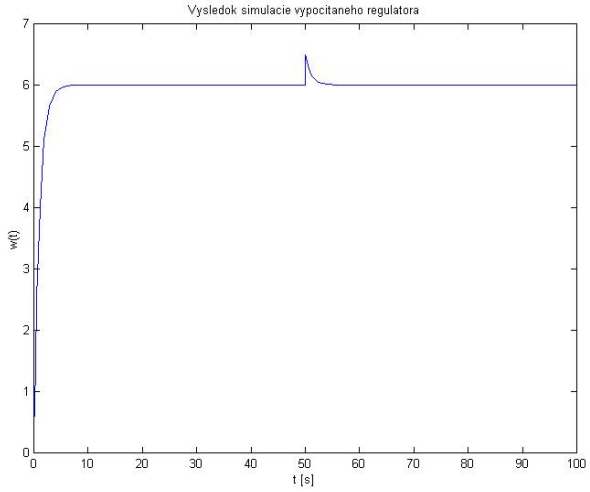
Výsledok po odoslaní je



Druhý krát som si zvolil dĺžku simulácie 500 s, počiatočný čas žiadanej veličiny 0 s, konečná veľkosť žiadanej veličiny 6, čas, v ktorom sa objaví porucha 50 s, veľkosť tejto poruchy 0.5, iný čitateľ a menovateľ prenosu nášho spätnoväzbového modelu i iné váhové keoficienty



a výsledok je



6 ZÁVER

V tejto práci som stručne priblížil problematiku inštalácie ako i konfigurovania WWW serveru Apache ako i Matlab Web Serveru. Bližšie som sa zaoberal využívaním vzdialeného prístupu užívateľov k modelom systémov prostredníctvom internetu, ktorý umožňuje modelovanie dynamických vlastností systémov. Rozpracoval som postup pri vytváraní takýchto modelov zahŕňajúci vytváranie vstupných HTML formulárov, vytváranie riadiacich funkcií a nakoniec vytvorenie výstupnej HTML šablóny. Na základe takto vymedzených postupov som vytvoril dynamický návrh LQ regulátora 1 DoF a 2 DoF systému s príkladmi. Na prezentáciu prechodových charakteristík jednotlivých veličín som využil grafy zobrazujúce sa prostredníctvom výstupnej HTML šablóny. Záverom možno konštatovať, že sa mi podarilo vytvoriť funkčné modely jednotlivých systémov, ktoré je možné ovládať prostredníctvom internetového prístupu.

Conclusion

In this review I tried to briefly explain the installation and configuration of WWW server Apache and also Matlab Web Server. More extensively I debated using remote access of users to models of systems via internet, which allows modelling of dynamic properties of systems. I described the technique of creating such models concerning creating input HTML forms, creating control functions and finally output HTML templates. Following these methods I created interactive models design of LQ regulator of 1DoF and 2DoF sysems with examples. The results – transmission functions I presented in the graphical way – by showing graphs at output HTML template. I can summarize that I succeed by creating functioning models of particular systems. All of these can be control by remote internet access.

Zoznam použitej literatúry

[1] Dokumentácia k Matlab Web Serveru. dostupné na www.mathworks.com

[2] Ľ. Čirka: Phd, Bratislava, 2003

[3] Dokumentácia k HTML www.jakpsatweb.cz

[4] Ľ. Čirka a kol.: Youla-Kučera parametrisation approach to LQ tracking and disturbance rejection problem, 2005